Universal Serial Bus
Device Class Definition
for
Communication Devices

0.7 Draft Revision
February 2, 1996

Scope of this Revision

The 0.7 release candidate of this definition is intended for industry review.

Revision History

Revision	Date	Filename	Comments
0.7	2/2/96	usb_cdc7.doc	Draft for industry review.

Contributors

Shelagh Callahan	Intel
Joe Decuir	Microsoft Corporation
Randy Fehr	Northern Telecom
Nathan Peacock	Northern Telecom
Dave Perry	Mitel Corporation

Please send comments via electronic mail to us@mitel.com

Table of Contents

List of Tables

1. Introduction

The Communication Device Class defines a general purpose mechanism that can be used to enable all types of communication services on the Universal Serial Bus (USB).

1.1 Scope

Given the broad nature of communication equipment, this document does not attempt to dictate how all communication equipment should use the USB. Instead, it attempts to define an architecture that is capable of supporting any communication device. The document's initial releases (versions earlier than 2.0) are limited to the definition of connectivity to telecommunication services—devices that have traditionally terminated an analog or digital telephone line.

It is not the intent of this document to redefine existing standards for connection and control of communication services. The Communication Device Class defines a means for a device and host to determine which protocols they wish to use. Where possible, existing protocols are used and transport of these protocols are merely enabled by the USB through the definition of the appropriate descriptors, interfaces, and requests.

1.2 Purpose

This document provides information to guide implementers in using the USB logical structures for communication devices. This information applies to manufacturers of communication devices and system software developers.

1.3 Related Documents

Universal Serial Bus Specification, 1.0 final draft revision (also referred to as the *USB Specification*)

ANSI/TIA-602, *Serial Asynchronous Automatic Dialing and Control*

ANSI/TIA-617, *In-Band DCE Control*

TIA IS-101, *Voice Control Interim Standard for Asynchronous DCE*

ITU V.25ter, *Serial Asynchronous Automatic Dialing and Control*

ITU Q.921, *ISDN User-Network Interface Data Link Layer Specification*

ITU Q.931, *ISDN User-Network Interface—Layer 3 Specification for Basic Call Control*

1.4 Terms and Abbreviations

ASVD	Analog Simultaneous Voice and Data, a modem that mixes data and audio.
AT command set	A telecommunication device control protocol. For details, see TIA-602 or V.25ter.
BRI	ISDN Basic Rate Interface, consisting of one D channel and two B channels.
DCE	Data Circuit Terminating Equipment. For example, a modem or ISDN TA.
DSVD	Digital Simultaneous Voice and Data, a modem that mixes data and digitized voice.
DTE	Data Terminal Equipment. For example, a PC.
GOI	Global Object Identifier. An X.208-formatted unique object identification string.

ISDN	Integrated Services Digital Network.
ITU	International Telecommunications Union (formerly CCITT).
POTS	Plain Old Telephone Service. See PSTN.
PRI	Primary Rate Interface, consisting of one or two D channels and up to 30 B channels.
PSTN	Public Switched Telephone Network.
TA	Terminal Adapter, the equivalent of a modem for ISDN.
TIA	Telecommunications Industry Association.
UART	Universal Asynchronous Receiver Transmitter
USB	Universal Serial Bus.
Video phone	A device that simultaneously sends voice and video with optional data.

2. Management Overview

Several types of communication devices can benefit from the Universal Serial Bus. This document provides models for the telecommunication subset of devices such as telephones, modems and ISDN terminal adapters. It describes:

- Relevant USB logical structures.
- How to assemble those logical structures into recommended configurations.
- The message structure for the CDC Control Interface. For a complete definition of this interface, see section 3.3.1 of this document.
- Commands and requests for the CDC Control Interface.
- Implementation examples of communication devices such as a basic telephone, modem, and ISDN Terminal Adapter.

3. Functional Characteristics

This section describes the functional characters of the Communication Device Class, including:

- Device operation
- USB endpoints
- How to construct interfaces from endpoints
- How to construct configurations from interfaces

3.1 Device Operation

This section describes in general terms how a device that belongs to the Communication Device Class (CDC) will operate. While it is usually simple to define the physical connections and interfaces of communication devices, it is much more difficult to define the types of data that move over these interfaces. For example, an ISDN Basic Rate Interface (BRI) can easily be defined in physical terms by its two 64 Kb B channels and one 16 Kb D channel. However, the nature of the data or media carried over those channels can change between and during calls.

Because of the static nature of the physical connections, this document describes them in terms of USB device, interface, and endpoint descriptors. The data that moves over the physical interfaces is dynamic in nature, and is therefore defined in terms of messages between the device and host over the default endpoint and an interrupt pipe, called the CDC Control Interface. The device can use these messages to inform its device driver when an interface is available and the format of the data to be transferred over that interface. The device driver can also use messages to retrieve information about data formats for an interface and select a data format when more than one is available.

Communication devices may present data to the host in a form defined by another device class, such as audio or human input. In order to allow the appropriate class driver to manage that data, the host is presented with an interface(s) that acts like one defined by that class. The interface that is required may change based on events initiated by the user or the network during a communication session—for example, the transition from DSVD Data mode to DSVD mode. Because the data presented by the communication device is so dynamic, this document describes a mechanism that allows an adaptive class device driver to query the communication device for information about an interface it has presented to the driver.

3.2 Endpoint Definitions

The communication device framework uses six endpoint definitions, described in Table 3-1.

Table 3-1: USB Communication Device Class Endpoints

Type	Interface usage	Reference
Default	CDC control and USB control	3.3.1
Interrupt	CDC control and interrupt feedback	3.3.1
Read isochronous	Isochronous data transfers	3.3.2
Write isochronous	Isochronous data transfers	3.3.2
Read bulk	Bulk data transfers	3.3.3
Write bulk	Bulk data transfers	3.3.3

The Default endpoint is endpoint 0, which exists on all USB devices. It is mentioned here because it is used for all communication control in addition to its regular USB duties. The Interrupt endpoint is expected to exist on all CDC devices, and is used for all notifications and synchronization-related interrupts. The usage of the isochronous and bulk endpoints is determined at runtime.

3.3 Interface Definitions

Each configuration consists of one or more interfaces and has a unique class descriptor that defines the capabilities of the interfaces included in the configuration. Using the information in the class descriptor, a driver can determine the capabilities of the device in a particular configuration, and can set the appropriate parameters of the configuration to enable transport of a particular data or media type.

The following sections describe the standard interfaces defined by the communication device class.

3.3.1 CDC Control Interface

CDC Control information can be divided into the following categories:

- DCE control for local devices.
- End-to-end control. For example, H.245 for H.324 systems.
- Extended PC-modem control (currently under study in ITU and TIA).

The CDC Control Interface is constructed from the default endpoint and an interrupt endpoint.

The default endpoint is intended for the transport of information which is not very time-sensitive or bandwidth-intensive. For example, the AT command with no additional arguments returns an ‘OK’ message immediately. The command expects an immediate response and does not have to wait for the device to perform any action before returning a known short-length response. Because of these properties, the AT command and its response are communicated over the default endpoint. Conversely, the AT&V command requires the device to deliver a table of its current settings, which may be several screens in

length. In this case, it is appropriate for the response to be delivered on the interrupt endpoint to avoid the risk of overloading the default endpoint, which is used to transport many other types of USB and class-specific commands.

The generic method of addressing endpoints within a command—used in the preceding example to inform the device which endpoint to use for its response—is defined in section 7.2.2.

The following functions are implemented on the default endpoint:

- Basic telephony functions such as placing and answering calls and detecting on/off hook.
- Delivery of device-control commands such as AT commands or Q.931.
- Reporting solicited command responses and result codes.
- Setting logical UART functions such as port rate, character format, and DTR / RTS states.
- Reading logical UART status such as port configuration and DSR / CTS / RLDS / RI status.

The following functions are implemented on the interrupt endpoint:

- Device-generated events such as on/off hook state transitions, legacy control lead changes, and user initiated device key presses.
- Network events such as incoming call notifications.
- Notifications of the interface availability and data formats.

3.3.2 Isochronous Data Interface

Isochronous pipes are used for data that meets the following criteria:

- Constant bit rate
- Real time communication that requires low latency

In general, isochronous endpoints can be used where raw information (either sampled or direct) from the network is sent to the host for further processing and interpretation. For example, an ISDN TA could use isochronous pipes for transport of each bearer channel as well as for the data channel. In this case, the host system would be responsible for decoding the D channel HDLC and any other lower level protocols.

The type of media encoding to be used is specified via messaging over the CDC Control interface when the host activates a media interface or the device requests that a media interface be activated. The bandwidth of the pipe is defined by the endpoint descriptors and can be changed by selecting an alternate interface of an appropriate bandwidth. Note that an alternate interface must be defined for every possible bandwidth. This is most evident in the case of ISDN BONDING where an interface of either 56/64 kb/s or 112/128 kb/s could be selected based on the session requirements. Note also that this example would not apply to the MP (Multipoint PPP) case, because the data is not required to be bit-synchronous and also would likely be delivered via a bulk interface.

3.3.3 Bulk Data Interface

The bulk data interface provides a method of moving potentially bursty, less time-sensitive data. Examples of bulk data include X.25 packet data on an ISDN D-channel, or data to or from a traditional modem. A bulk data interface could also be used for moving audio over the USB where real-time interaction is not required, as in the case of a voice mail application.

For telephony devices, bulk endpoints are suitable for end user data carried on ISDN D channels, bearer data such as character streams including facsimile, and compressed video. In all cases, the choice of bulk or isochronous is based on the nature of the data and not on the physical network connection.

Bulk endpoints are also suitable for some kinds of audio data, particularly data received from or sent to a voice codec. Voice codecs are distinguished from audio codecs or waveform codecs by the characteristics in Table 3-2.

Table 3-2: Characteristics of Audio, Waveform, and Voice Codecs

	Audio Codec	Waveform Codec	Voice Codec
Example application	44.1Ks/s CD audio	G.721 ADPCM	G.723
Data rate	High: 705600 bit/s	Medium: 32000 bit/s	Low: 5500 bit/s
Granularity	Sample	Sample	15-20ms of samples
Inherent delay	Sample	Sample	15-20ms
Complexity	Low	~1MIP	~20MIPS
Model	Sample	Compress dynamic range only	Vocal tract

Given these characteristics, bulk transport may be preferable to isochronous for transfer of compressed audio — bulk provides guaranteed delivery and for low data rates might provide adequate performance even on a congested bus.

3.4 Configurations

Particular USB communication device configurations are constructed from the interfaces described in previous sections. The following examples illustrate how interfaces can be combined to implement configurations for basic telephony, modem, and ISDN devices.

3.4.1 Basic Telephony Configurations

This section defines two examples of telephony configurations: a basic telephone and a telephone with audio. The minimum requirement for this type of device is a configuration with a single CDC Control Interface. The most basic audio-capable telephone is constructed by adding an Isochronous Data interface for audio transmission and reception. A more advanced configuration could optionally have two **Local Data** interfaces (handset and microphone/speaker), and one **Line Data** interface. In this case the Line Data interface could be the raw linear data as sampled from the network. The responsibility for demodulation and interpretation of this data would lie within the host at the application level (i.e. NSP Modem). Any vendor-specific interfaces would follow.

Table 3-3: Telephone Configurations

Example Configuration	Interface	Reference Section	Description
Basic telephone	CDC Control	3.3.1	USB telephone.
Audio telephone	CDC Control	3.3.1	USB control, device control.
	Bidirectional Isochronous Data	3.3.2	I/O for line audio. Additional identical interfaces could be added for local audio.
Audio/data telephone	CDC Control	3.3.1	USB control, device control.
	Bidirectional Isochronous Data	3.3.2	I/O for line audio. Additional identical interfaces could be added for local audio.
	Bidirectional Isochronous Data	3.3.2	I/O for line data.

A communication device that supports audio type media streams over its interfaces can use the selected host-device command language to indicate which voice or audio coding formats it supports (for example, IS-101 for voice modems).

3.4.2 Modem Configurations

This section defines two examples of modem configurations: legacy modem and multimedia modem. The first configuration covers today's single-media modems for data, fax, voice, VoiceView*, and text phones. The second configuration covers new multimedia modems, such as ASVD modems (for example, ITU V.61), DSVD modems (for example, DSVD SIG 1.1) and video phones (ITU H.324).

Table 3-4: Example Modem Configurations

Example Configuration	Interface	Reference Section	Description
Legacy modem	CDC Control	3.3.1	USB control, device control.
	Bidirectional Bulk Data	3.3.3	I/O for demodulated, decompressed modem data.
DSVD modem	CDC Control	3.3.1	USB control, device control, synchronization control and feedback, media control, and interface control.
	Bidirectional Bulk Data	3.3.3	I/O for demodulated, decompressed modem data.
	Bidirectional Isochronous Data	3.3.2	I/O for uncompressed audio.
Multimedia modem	CDC Control	3.3.1	USB control, device control, media control, synchronization control and feedback, media and interface control.
	Bidirectional Bulk Data	3.3.3	I/O for demodulated, decompressed Modem data.
	Bidirectional Isochronous Data	3.3.2	I/O for uncompressed audio.
	Bidirectional Isochronous Data	3.3.2	I/O for uncompressed video (for example H.263). A Bidirectional Bulk Data interface could optionally be used to transport compressed video information if the compression/decompression resource was resident on the host.

Most of today's modem type devices—single-media or multimedia—contain various types of media processing resources such as compression engines (for example, V.42bis) or audio/video codecs. Given the projections of increased processing power for future host systems and the availability of appropriate media transport to and from the host (that is, the USB), it is likely that various models of media processing will emerge that do not rely solely on the device for these resources. In this case, where media processing resources are located arbitrarily within the system (for example, V.42bis on the host and V.34 on the device), interface choices for various media types could vary. For example, if a device developer chose to include an MPEG2 codec in a device, a bi-directional isochronous interface may be more appropriate for transport of the video stream. Conversely, if the codec is not in the USB device, a bi-directional bulk interface would be more appropriate.

The processing required for some types of media streams is asymmetrical in nature. For example, MPEG2 decompression is trivial by today's standards, while compression requires substantial processing resources. In light of this fact, it may be appropriate to configure an interface with the appropriate asymmetry. Continuing the MPEG example, a device that relies on host-based decompression and device-based compression would choose an interface that consists of an isochronous endpoint for video in the host-to-device direction and a bulk endpoint for the device-to-host direction.

Some examples of the bandwidth implications of device vs. host based media stream processing are listed below. USB bandwidth is expressed in bytes per millisecond (B/ms).

- Typical performance of V.42bis is 3-4:1 on data streams; 33.6 kb/s V.34 data could unpack to 16.8 B/ms.
- G.723 voice codec (5.5 - 6.5 Kb/s) could be unpacked to 11 kHz audio by the modem (22 B/ms).
- H.263 compressed video is in the 2.5 B/ms range, but if H.263 is decompressed, a typical bandwidth is approximately 96 B/ms.

3.4.3 ISDN Device Configurations

The two broad categories of ISDN services are Basic Rate Interface (BRI) and Primary Rate Interface (PRI). BRI universally consists of two 64 kb/s bearer channels and one 16 kb/s data channel, while PRI is regionally defined. In North America and Japan, PRI is made up of twenty-three 64 kb/s B channels and one 64 kb/s D channel for a total of 1.544 kb/s (T1 or DS-1 transmission rate). The rest of the world generally uses thirty B channels and two 64 kb/s D channels (E1 transmission rate of 2.048 Mb/s).

The USB is intended to be an end user-oriented, desktop device connection bus, so the uses of ISDN in this document are limited to those which are likely to occur at a typical desktop.

Several options exist for configuration of an ISDN device. In the simplest case, both the B channels and D channels can be delivered to the host in raw form. In this case, the USB would obviously run out of endpoints before an E1 rate PRI would, thereby necessitating the aggregation of bearer channels.

The aggregated B channels would be further de-multiplexed by the host at an application level. If isochronous transport is selected for the D channel(s), low level (HDLC) processing would also be the responsibility of the host.

An interesting set of devices could be implemented which would further process the PRI stream and return various media streams to the host. Interface selection would be based on the nature of the media, as described in the above sections. For example, a fractional PRI USB interface might use MP+ to associate a number of B channels together to be used as a WAN connection. In this case, the nature of the media would indicate that a bi-directional bulk interface is appropriate to transport the data to the host. The remaining B channels could be used for telephone-type audio sessions and activated as bi-directional isochronous interfaces, keeping in mind that endpoints on USB are a finite resource.

In general, B channels can be associated together at a bit-synchronous level and passed to the host via isochronous pipes, or the media being carried on the B channels can be decoded and passed to the host via an appropriate interface.

An ISDN BRI device may commonly be combined with a basic or audio-enabled telephone functionality, so that local audio channels are included as an optional element in the configuration.

Table 3-5: Example ISDN Configurations

Example Configuration	Interface	Reference Section	Description
ISDN BRI terminal adapter	CDC Control	3.3.1	USB control, device control, end-to-end signaling, call control (all D channel signaling).
	Bi-directional Isochronous Data	3.3.2	I/O for B1 media.
	Bi-directional Isochronous Data	3.3.2	I/O for B2 media.
ISDN telephone with terminal adapter	CDC Control	3.3.1	USB control, device control, end-to-end signaling, call control (all D channel signaling).
	Bi-directional Isochronous Data	3.3.2	I/O for B1 media.
	Bi-directional Isochronous Data	3.3.2	I/O for B2 media.
	Bi-directional Isochronous Data	3.3.2	I/O for local audio (that is, speaker/microphone).
North American ISDN PRI interface	CDC Control	3.3.1	USB control, device control.
	Bi-directional Bulk Data	3.3.3	End-to-end signaling, call control (all D channel signaling).
	Bi-directional Isochronous Data	3.3.2	I/O for B1-B6. Further de-multiplexing required by host.
	Bi-directional Isochronous Data	3.3.2	I/O for B7-B23. Further de-multiplexing required by host.

Table 3-6: Example ISDN Configurations (continued)

Example Configuration	Interface	Reference Section	Description
ISDN PRI audio/data device	CDC Control	3.3.1	USB control, device control.
	Bi-directional Bulk Data	3.3.3	End-to-end signaling, call control (all D channel signaling).
	Bi-directional Bulk Data	3.3.3	B1 through B12 WAN data associated via MP+.
	Bi-directional Bulk Data	3.3.3	B13 through B18 WAN data associated via MP+ (second WAN connection).
	Bi-directional Isochronous Data	3.3.2	I/O for B19 audio media. Additional identical interfaces could be added for audio on remaining B channels.

For further study: mapping H.320 video conferencing to a USB ISDN BRI configuration.

4. Class-Specific Codes for Communication Devices

This section lists class-specific codes for the Communication Device Class, subclasses, and protocols. These values are used in the **bDeviceClass**, **bDeviceSubClass**, and **bDeviceProtocol** fields of device descriptors for this class.

This section also lists media type codes, which are used in the **bPurpose** field of a Notification of Interface Availability message. For details, see 7.2.4.

4.1 Communication Device Class Code

The class code for the Communication Device Class is COMMUNICATIONS (2). This code is assigned by the USB committee.

4.2 Communication Device Subclass Codes

This document defines the following subclass codes for the Communication Device Class:

Table 4-1: Communication Device Subclass Codes

bDevice code	Reference	Subclass Name
00		Reserved
01		Telephony Interface Device Subclass
02-7F		Reserved
80-FE	6.2.2	Vendor-specific
FF		Reserved

Only one subclass code is currently defined for the Communication Device Class: the Telephony Interface Device subclass, which is used to describe devices that link a computer to a telephone network. When additional communication devices are supported (for example, low-speed LAN devices such as a USB wireless LAN adapter), other subclasses will be added. Ranges of subclass codes are set aside for vendor specific devices and future USB designated subclasses.

Any parties interested in defining device subclasses that are not covered in this document should contact the USB committee.

4.3 Control Protocol Codes

A Communication Device Control protocol is used by the USB host to control communication functions in the device. This specification defines code values for some standard control protocols. It also reserves codes for other standard or vendor-specific control protocols.

Protocol codes apply at both the class level and the interface level. For example, an ISDN TA/modem hybrid device may direct AT commands to control a modem data interface while directing raw Q.931 messages to control a native ISDN interface. Both control protocols use the CDC Control Interface for transport and each command is addressed to the appropriate interface for that command.

Table 4-2: Control Protocol bDeviceProtocol Codes

bDeviceProtocol Code	Reference Document	Description
00		Undefined
01	V.25ter	Common AT Commands (also known as “Hayes* compatible”)
02	V.25bis	Alternative PSTN modem command set, (sync modems, some ISDN TAs)
04	V.120	Serial ISDN Terminal Adapter Control
08	V.ib	In-Band DCE control, (superset of TIA-617)
10	Q.931	ISDN TA control (implies the use of Q.921 as transport)
20		Reserved
40	6.2.1	Other standard DCE control protocol
80	6.2.2	Manufacturer-proprietary DCE control protocol

Note: There is some discussion about whether device protocol codes should be stored in a bitmap. It is currently proposed that the device protocol be a list of codes which would be returned by an appropriate command/request structure. In this way, the host would issue a **GetDeviceProtocol** command for an interface or device and the device would return a list of protocol codes it supports on that interface or device.

4.4 Media Type Codes

A Communication Device may exchange data in a variety of formats. This section defines codes for some standard media formats. It also reserves codes for other standard or vendor-specific media formats.

This document references several subclasses of media type codes. These values are used in bits 5-0 of the **bPurpose** field in the Notification of Interface Availability request. (See 7.2.4.)

Table 4-3: Media Type bPurpose Codes

Media Type	bPurpose Code (hex)	Reference Section	Description
None	0		Undefined
Synchronous data	1	4.4.1	Simple bit streams or synchronous frame streams
Character data	2	4.4.1	Character streams
PCM audio	3	See note below	High-fidelity PCM audio
Waveform-coded audio	4	4.4.2	Toll quality PCM or ADPCM
Voice-coded audio	5	4.4.2	Comm quality, voice-coded audio
Compressed video	6	4.4.2	Telecom quality compressed video
End-to-end control	7		ISDN or conferencing end-to-end control channel
Reserved	8-F		

Note: PCM audio descriptions are incorporated by reference from the Audio Device Class Specification.

The values defined in the following sections are used in the **bProtocol** field in the Notification of Interface Availability request. (See section 7.2.4.)

4.4.1 Protocol Codes for Data Endpoints

Data endpoints are sorted into three groups:

- Synchronous data: the data is simple bits-to-bits, or it is synchronous on the network interface, but the communication device supports conversion from an octet stream to a synchronous protocol.
- Character data: the data is assumed to be 7 or 8-bit characters. The DCE may provide character framing bits and other information.
- End-to-end control protocols: the data is control information that is exchanged with the remote network terminal and/or intermediate points.

Table 4-4: Data Endpoint bProtocol Codes

Endpoint Purpose	bProtocol Code (hex)	Type	Description
Synchronous data	00	None	Undefined
	01	Simple	Bits-in-bits-out
	02	HDLC	ISO-3309 HDLC framed mode
	03	Asynch HDLC	ISO-3309 frame tunneling mode
	04	X.25	Public data network, data link layer
	05	Q.921	ISDN data link layer
	06-9F	-	Reserved
	A0-FE	6.2.2	Manufacturer synchronous data protocol
	FF		Reserved
Character data	00		Undefined
	01	V.4	The DCE produces and consumes start and stop bits
	02	V.14	The DCE produced and consumes start and stop bits, with rate matching
	03	V.42	The DCE implements layer 2 error control protocol
	04	V.42bis	The DCE implements layer 2 error control and data compression
	05	PPP	The DCE implements PPP
	06-9F		Reserved
	A0-FE	6.2.2	Manufacturer character data protocol
	FF		Reserved

Table 4-5: Data Endpoint bProtocol Codes (continued)

Endpoint Purpose	bProtocol Code (hex)	Type	Description
End-to-end control	00		Undefined
	01	Q.931	ISDN control protocol
	02	H.245	Multimedia Terminal control protocol
	03-9F		Reserved
	A0-FE	6.2.2	Manufacturer end-to-end control protocol
	FF		Reserved

This specification does not specifically address the format of messages: images or files. This is expected to be implemented via the character or synchronous data modes. For example:

- Image data: T.4, T.6, T.81 (JPEG), T.82 (JBIG)
- File data: T.434, MIME

4.4.2 Protocol Codes for Audio/Voice Media Endpoints

Audio/voice media endpoints are sorted into three groups:

- Audio: These are high fidelity and high data rate. The **bProtocol** codes are specified in the Audio Device Class Specification, section tbd.
- Waveform: Telecom waveform coding protocols.
- Voice: Telecom voice coding schemes that depend on human vocal tract models.

Table 4-6: Audio/Voice bProtocol Codes

Endpoint Purpose	bProtocol Code (hex)	Type	Description
Waveform	00		Undefined
	01	G.711a	A-law encoded PCM
	02	G.711u	Mu-law encoded PCM
	03	G.721	32kbit/s 3.1kHz bandwidth ADPCM
	04	G.722	64kbit/s 7kHz bandwidth ADPCM
	05	IMA	IMA ADPCM
	06-9F		Reserved
	A0-FE	6.2.2	Manufacturer character data protocol
	FF		Reserved
Voice coders	00		Undefined
	01	GSM 6.10	13.3kb/s cellular phone coder
	02	GSM 6.20	6.5kb/s cellular phone coder
	03	G.723	5.5/6.5kbit/s voice coder (H.324)
	04	G.72X	8.5kbit/s voice coder (V.dsvd)
	05-9F	-	Reserved
	A0-FE	6.2.2	Manufacturer end-to-end control protocol
	FF		Reserved

TBD: Look up the codec standards used in North American cellular.

4.4.3 Protocol Codes for Compressed Video Endpoints

This document addresses compressed video formats used in conjunction with the PSTN or ISDN. Other data formats may be too high to accommodate on the USB.

Table 4-7: Compressed Video bProtocol Codes

Endpoint Purpose	bProtocol Code (hex)	Type	Description
Compressed video	00		Undefined
	01	H.261	Video codec for ISDN, H.320
	02	H.263	Video codec for PSTN, H.324
	03-9F		Reserved
	A0-FE	6.2.2	Manufacturer character data protocol
	FF		Reserved

4.5 Synchronization Type Codes for Communication Devices

When a Communication Device uses isochronous pipes, it is necessary to use some method of synchronization. This section defines codes for some standard synchronization types. It also reserves codes for other standard or vendor-specific synchronization types.

This document defines several synchronization type codes. These values are used in bits 6-7 of the **bPurpose** field of the Notification of Interface Availability request (see 7.2.4).

Table 4-8: Sync Type Codes

Sync Type Value	Type	Description
00	None	No synchronization used.
01	Asynchronous	See section 5.10.4, “Isochronous Devices, “ in the USB Specification.
02	Synchronous	See section 5.10.4, “Isochronous Devices, “ in the USB Specification.
03	Adaptive	This source or sink can adapt to the sync method of the source or sink it is communicating with.

Note: The Audio Device Specification also explains synchronization methods well and the same information will be referenced properly or included here.

5. Standard USB Descriptor Definitions

This section defines standard USB descriptors for the Communication Device Class.

5.1 Device Descriptor

The device descriptor for the Communication Device Class is as defined in section 9.7.1 of the USB specification, with the following qualifications:

- The **bDeviceClass** field is / can be set to the Communication Device Class code.
- Initially, the **bDeviceSubClass** is set to the Telephony Interface identifier code (see Table 4-2).
- The **bDeviceProtocol** field defines the call control language used on the CDC Control Interface. Recall that different protocols can be targeted to controlling specific interfaces (for details, see section 4.3). If the protocol is not defined for the bitmap, then "Other" can be selected, and the supported protocols are returned following the device descriptor using one or more protocol descriptors.

Table 5-1: Device Descriptors

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor in bytes.
1	bDescriptorType	1	Constant	DEVICE Descriptor Type.
2	bcdUSB	2	BCD	USB Specification Release Number in Binary-Coded Decimal (for example, 2.10 is 0x210). This field identifies the release of the USB Specification that the device and its descriptors are compliant with.
4	bDeviceClass	1	Constant	COMMUNICATIONS (2). This code is assigned by the USB.
5	bDeviceSubClass	1	SubClass	See section 4.2.
6	bDeviceProtocol	1	Bitmap	Indicates the Call control protocols to be used in encapsulated requests on the default endpoint. See Section 4.3.
7	bMaxPacketSize0	1	Number	Maximum packet size for endpoint zero (only 8, 16, 32, or 64 are valid).
8	idVendor	2	ID	Vendor ID (assigned by USB).
10	idProduct	2	ID	Product ID (assigned by manufacturer).
12	bcdDevice	2	BCD	Device release number in Binary-Coded Decimal
14	iManufacturer	1	Index	Index of string descriptor describing manufacturer.

Table 5-1: Device Descriptors (continued)

Offset	Field	Size	Value	Description
15	iProduct	1	Index	Index of string descriptor describing product.
16	iSerialNumber	1	Index	Index of string descriptor describing the device's serial number.
17	bNumConfigurations	1	Number	Number of possible configurations.

5.2 Configuration Descriptor

The Communication Device Class uses the standard configuration descriptor defined in section 9.7.2 of the USB specification.

5.3 Interface Descriptors

The Communication Device Class uses the standard interface descriptor as defined in section 9.7.3 of the USB specification with the following qualifications:

- The **bInterfaceClass** field can be the interface code for either the Communication Device Class or any other appropriate class (for example, the Human Interface Class or Audio Class).
- If the **bInterfaceClass** field is the interface code for a Communication Device Class, the **bInterfaceSubClass** field must be Telephony Interface Device.
- If the **bInterfaceClass** field is the interface code for a Communication Device Class, the **bInterfaceProtocol** is a flag that indicates whether the responses to commands on the default pipe can be returned over the interrupt pipe of the CDC Control Interface.

Table 5-2: Interface Descriptors

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor in bytes.
1	bDescriptorType	1	Constant	INTERFACE Descriptor Type.
2	bInterfaceNumber	1	Number	Number of interface. Zero-based value identifying the index in the array of concurrent interfaces supported by this configuration.
3	bAlternateSetting	1	Number	Value used to select alternate setting for the interface identified in the prior field.
4	bNumEndpoints	1	Number	Number of endpoints used by this interface (excluding endpoint zero). If this value is zero, this interface only uses endpoint zero.

Table 5-3: Interface Descriptors (continued)

Offset	Field	Size	Value	Description
5	bInterfaceClass	1	Class	Interface Class code (assigned by USB) for the Communication Device Class.
6	bInterfaceSubClass	1	SubClass	See section 4.2.
7	bInterfaceProtocol	1	BOOL	Unused. Reset to 0.
8	iInterface	1	Index	Index of string descriptor describing this interface.

5.4 Endpoint Descriptors

The Communication Device Class uses the standard endpoint descriptors as defined in section 9.7.4 of the USB specification.

6. Class-Specific Descriptors

This section describes class-specific descriptors for the Communication Device Class.

6.1 Class-specific Configuration Descriptor

Note: This section is still under development. It requires much more review and should be treated as such.

This descriptor is used to define the capabilities of the device from a communications point of view for each configuration.

-

Table 6-1: Device Class Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor in bytes.
1	bNetworkConnections	1	Number	Number of network connections (See note, below).
2	bDeviceProtocol			The method of device control via the CDC Control Interface.

Note: How the network connections are counted depends on how the device handles the connections. For example, an ISDN BRI could count as one network connection (bonded B channels), two network connections (separate B channels) or three network connections (separate B channels and an X.25 D channel), depending on how the device intends to give access to the network. Typically, a POTS connection counts as a single network connection.

6.2 Protocol Descriptor

This document reserves **bProtocol** codes to specify two class-specific string descriptors:

- Global Object Identifiers (value 40 hex)
- Manufacturer Specific Identifiers (value 80 hex)

If the Device Class Descriptor uses one of these as a **bProtocol** value, the Device Class Descriptor will be followed by one or more Protocol Descriptors that describe the protocols to be used.

Table 6-2: Protocol Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	6	Size of this descriptor in bytes
1	bProtocolID	5	Number	Protocol to be used: 0x40 ISO Global Object Identifier 0x80 Manufacturer-specific Identifier

6.2.1 Global Object Identifiers (GOI)

A manufacturer may implement a standard protocol for media or control that is not listed in this document. To indicate this choice, the manufacturer should use a **bProtocol** value of 0x40 where applicable, and use a string descriptor that consists of a Global Object Identifier (GOI) to identify the protocol being used.

A GOI consists of a string of characters that uniquely identifies an object. The string is encoded in ASN.1 (ITU X.208, or ISO equivalent), using standard encoding rules. The string will be a set of delimited numbers, beginning with either an international root code (for example, ISO, ITU, or IEEE) or a national root (for example, a country code) and followed by subsidiary identifiers. For example:

insert examples of GOIs defined for some standard objects referenced in this document.

6.2.2 Manufacturer-Specific Identifiers

A manufacturer may implement a proprietary protocol for media or control. To indicate this choice, the manufacturer should use a **bProtocol** value of 0x80 where applicable, and use a string descriptor to identify the protocol being used. The **idVendor** field could be used to distinguish between vendors.

7. Requests

Note: This section is still under development. It requires much more review and should be treated as such.

7.1 USB Standard Device Requests

The Communication Device Class supports the standard requests defined in section 9.6 of the USB Specification.

7.2 Class-specific Requests

The Communication Device Class supports the following class-specific requests:

Send Encapsulated Command
Get Encapsulated Response
Report Format (encapsulated protocol message)
Notification of Interface Availability
Select Interface Protocol Command
Get Interface Command

7.2.1 Send Encapsulated Command

This request transports encapsulated commands as defined by the **bProtocol** field in the device class descriptor.

Table 7-1: Encapsulated Command Format

Offset	Field	Size	Value	Description
0	bRequestType	1	001xxxxx	Characteristics of request: D7=0 Transfer is host to device D6..5=01 Type is Class D4..0=xxxxx See USB Specification
1	bRequest	1	cSend_ Encapsulated _Command	Request code for this request.
2	wValue	2	Word	See Table 4-2
4	wIndex	2	Word	The interface address of the recipient is contained in the lower byte of this field. The upper byte is set to 0
6	wLength	2	N	Protocol-dependent (N)

7.2.2 Get Encapsulated Response

This optional command requests an encapsulated report. The result is returned on either the default pipe or the interrupt pipe of the CDC Control Interface, depending on the nature of the response. Normally an encapsulated report would be relatively long and would therefore be appropriate to return on the interrupt pipe.

Table 7-2: Encapsulated Command Response Format

Offset	Field	Size	Value	Description
0	bRequestType	1	101xxxxx	Characteristics of request: D7=1 Transfer is device to host D6..5=01 Type is Class D4..0=xxxx See USB Specification
1	bRequest	1	cGet_Encapsulated_Response	Request code for this request.
2	wValue	2	word	Protocol ID.
4	wIndex	2	word	The interface address of the recipient is contained in the lower byte of this field. The upper byte is set to 0.
6	wLength	2	N	Protocol-dependent (N)

7.2.3 Report Format (Encapsulated Protocol Message)

Encapsulated protocol messages are sent to the host through the interrupt pipe in the form of reports. Reports may also be requested (polled) through the control pipe.

The report format is composed of a class code byte, a protocol ID field, a two-byte length field, and the encapsulated message (data). The class code byte is used to allow other report formats from other classes (for example, HID) to exist on the same pipe. The Protocol ID field allows a client to implicitly route the message to the appropriate entity. The length field is two bytes to allow long protocol messages to be encapsulated (for example, Q.931). The length field specifies the length of the data in bytes, excluding the encapsulation header.

Table 7-3: Encapsulated Report Format

Offset	Field	Size	Value	Description
0	bClassCode	1	Class Code	Assigned by USB
1	bRequestType	1	101xxxxx	Characteristics of request: D7=1 Transfer is device to host D6..5=01 Type is Class D4..0=xxxx See USB Specification
2	bRequest	1	cEncapsulated_Report	cEncapsulated_Report Code = 1
3	wIndex	2	Word	The interface address of the sender is contained in the lower byte of this field. The upper byte is set to 0.
5	wLength	2	N	Protocol-dependent (N)
7	Data	n		n bytes of data

7.2.4 Notification of Interface Availability

This section is a **first draft** of a proposal on dynamic management of interfaces and publication of interface availability to other potential sources or sinks (for example, an audio driver).

Notification of Interface Availability is an interrupt message that indicates that an interface(s) is available for use. It presents the IDs of the interfaces that are available, and how to use the interface (such as the protocol and data types). This message can also be used to indicate that an interface is being closed down by specifying the **bPurpose** as ***none***. The Class Code byte is used to determine the class that this endpoint will belong to (that is, the interface may belong to the CDC, audio, or HID class). If the class code is not the CDC class code, this interface can receive the **GetClassInfo** message and return the appropriate information so that the appropriate class device driver can use it. (See section #.#, **GetClassInfo**.)

Note that the device can only request that the host activate or deactivate an interface. It is the responsibility of the host to carry out or deny the request.

Table 7-4: Notification of Interface Availability

Offset	Field	Size	Value	Description
0	bClassCode	1	Class Code	Assigned by USB
1	bRequestType	1	101xxxxx	Characteristics of request: D7=1 Transfer is device to host D6..5=01 Type is Class D4..0=xxxx See USB Specification
2	bRequest	1	cInterface_Availablilty	cInterface_Availability Code = 2
3	wIndex	2	Word	The interface address of the sender is contained in the lower byte of this field. The upper byte is set to 0.
5	wLength	2	Number	Length of data to follow.
7	bPurpose	1	Number	See Table 4-3.
8	bProtocol	1	Number	See Table 4-4,Table 4-6, and Table 4-7.
9	bSync	1	Number	See Table 4-8.
…	…			
n-1	bProtocol	1	Number	
n	bSync	1	Number	

7.2.5 Select Interface Protocol

Select Interface Protocol is a control pipe message that selects an interface. The contents of this message are based on the contents of a Notification of Interface Availability. The only valid values for the **bProtocol** field are values from a Notification of Interface Availability for that interface.

This command is distinct from the command that opens an interface or endpoint. This message just indicates to the device what protocol is to be used.

Table 7-5: Select Interface Protocol Response Format

Offset	Field	Size	Value	Description
0	bRequestType	1	10100001	Characteristics of request: D7=0 Transfer is host to device D6..5=01 Type is Class D4..0=00001 Recipient is interface
1	bRequest	1	cSelect_Interface	Request code for this request.
2	wValue	2	Word	bSyncType/bProtocol.
4	wIndex	2	Word	The interface address of the recipient is contained in the lower byte of this field. The upper byte is set to 0.
6	wLength	2	0	

7.2.6 Get Interface

Get Interface is used to retrieve the existing settings of an interface. It returns the same information as the Notification of Interface Availability, with the addition of a field for the currently selected protocol.

Table 7-6: Get Interface Command Format

Offset	Field	Size	Value	Description
0	bRequestType	1	10100001	Characteristics of request: D7=1 Transfer is device to host D6..5=01 Type is Class D4..0=00001 Recipient is interface
1	bRequest	1	cGet_Interface	Request code for this request.
2	wValue	2	Word	bSyncType/bProtocol.
4	wIndex	2	Word	The interface address of the recipient is contained in the lower byte of this field. The upper byte is set to 0.
6	wLength	2	n	Depends on the number of protocols.
8	bPurpose Code	1	Number	See Table 4-3.

Table 7-7: Get Interface Command Format (continued)

Offset	Field	Size	Value	Description
9	bProtocol	1	Number	See Table 4-4,Table 4-6, and Table 4-7.
10	bSyncType		Number	See Table 4-8.
…			…	
n-1	bProtocol	1	Number	See bSyncType (Table 4-8), bProtocol (Table 4-4, Table 4-6, and Table 4-7).
n	bSyncType		Number	See Table 4-8.

Appendix A Communication Device Class Examples

This appendix provides some examples of typical communication device classes.

A.1 Basic Telephone

For compatibility with a broad range of PC software, a USB basic telephony device should provide the functions listed in section 3.1 of this document, controlled with the classic AT command set. This command set is documented in ANSI/TIA-602 and in ITU V.25ter. AT command signaling is implemented as class-specific encapsulated requests through the Basic Call control interface, over the default channel (see 7.2.1).

For compatibility with voice-aware PC software, a basic telephony device that also supports voice media stream(s) to the Local Data or Line Data interfaces should provide the AT+V voice media control commands defined in TIA IS-101, or equivalent (such as the AT#V command set, ITU V.60-series recommendation in development).

A.2 Modem

For compatibility with legacy PC software, and to facilitate the development of generic drivers, a USB modem should conform to the ANSI/TIA-602 standard. For common extended functions, the following standards are recommended:

- Modem identification: ITU V.25ter +G commands
- Data modems: ITU V.25ter (modulation, error control, data compression)
- Fax modems: ITU T.31 or T.32 +F commands (or TIA equivalents)
- Voice modems: TIA IS-101 +V commands
- General wireless modems: PCCA STD-101 +W commands (TIA PN-3499)
- Analog cellular modems: PCCA STD-101 Annex I (TIA PN-3499 Annex C)
- Digital cellular modems: TIA IS-99, TIA IS-135 or GSM 7.07 +C commands.
- Text phone modems: V.25ter, +MV18 commands.

See the ITU Supplement to V.25ter for a complete list of standard modem command sets.

Note: At this writing, common standards do not exist for DSVD or videophone modems, but they are under development in TIA TR-30 and ITU Study Groups 14 and 15.

Further, a USB modem may provide means to accommodate common functions performed on a 16550 UART. For details, see Appendix C, “Standard Serial Interface Circuit Emulation.”

A.2.1 Legacy Modem

TBD

A.2.2 Multimedia Modem

TBD

A.3 ISDN Devices

TBD

A.3.1 ISDN BRI TA

TBD

Appendix B AT Command Message Encapsulation

The most common telecommunication device control protocol is the AT command set. The basic AT command set is codified in ITU-T V.25ter, along with rules for extension and particular extensions for common functions. (These three components are also contained in TIA-602, TIA-615 and TIA IS-131 respectively.)

Section 7.2 specifies how to encapsulate standard communication device commands and responses. This section addresses rules for delivering AT command messages over the USB. In particular, it specifies which means of encapsulation to use for each type of AT command message.

V.25ter contains three types of messages: commands, information text and result codes. Command sets based on V.25ter use the same message types. The basic encapsulation rules are simple:

1. All commands from the DTE (the USB host) are encapsulated and transported to the device via the CDC Control Interface over the default pipe) (7.2.1).
2. All command responses that brief in nature and are available instantly should be encapsulated and returned via the default pipe (7.2.2).
3. All other command responses should be encapsulated and transported via the interrupt pipe (7.2.3).

Table B-1: AT Command Set Message Encapsulation

Message Type	V.25ter Reference	Encapsulation	Example(s)
Action Commands	5.2.5, 6.3.1, 6.3.5	7.2.1	ATA, ATD
Set Parameter	5.2.5, 5.3.2, 6.3.8, 6.4.1	7.2.1	ATS0=1, AT+MS=V34,1,,,
Read Parameter	5.2.5, 5.3.2, 6.3.8, 6.4.1	7.2.1	ATS0?, AT+MS?
Test Parameter	5.2.5, 5.3.2, 6.3.8, 6.4.1	7.2.1	AT+MS=?
Information Text If immediate response is ready	5.7.3	7.2.2	+MS:V34,1,300,28800 See below
Information Text Any delayed response	5.7.1	7.2.3	AT&V response See below
Intermediate Result Codes	5.7.1	7.2.3	+MCR: <carrier>, +MRR: <rate> +ER: <type>, +DR: (type>, +ILRR: <rate>

Table B-1: AT Command Set Message Encapsulation (continued)

Message Type	V.25ter Reference	Encapsulation	Example(s)
Final Result Codes (after immediate responses only)	5.7.1	7.2.2	OK, ERROR
Final Result Codes (after any delayed responses)	5.7.1	7.2.3	CONNECT, BUSY, NO CARRIER
Unsolicited Messages	6.3.4	7.2.3	RING

Note: The responses to parameter reads and parameter tests generate information text as well as result codes. In principle, the responses are available instantly. However, the device controller is not required to respond in the same USB frame cycle, since the timing requirements are very tight. Therefore, the communication device may respond with an indication to the controller to repeat the read or respond with the indication in the interrupt endpoint. (where is the reference in Chapter 9)

Common modems contain a small amount of EEPROM. The common AT&V command generates a read of the EEPROM, which is not instantaneous, and therefore must be delayed.

Appendix C Standard Serial Interface Circuit Emulation

To ease the migration of external communication devices from UART ports to USB ports, this section defines a means to model serial port functions on the USB default control endpoint. The most common example is the DB25 asynchronous port, described in ITU-T V.24 or TIA-232-E. (The 9-pin version of this interface is defined in TIA-574.)

C.1 Standard UART Functions

Three sets of functions are exported by common standard 16450 or 16550 UARTs:

- V.24 serial data, including serial data, data rate, and data format.
- V.24 out-of-band leads (DTR, DSR, etc.).
- PC internal bus interface functions (data registers, interrupts, FIFO control, additional leads.

V.24 in-band and out-of-band logic is exposed to the communication devices, and may need to be delivered via the USB. PC internal bus interface functions are hidden from the communication device; if needed, these functions should be implemented in the USB Communication Device Class Driver.

C.2 Serial Data Control

AT command modems use “autobauding” or hard straps to set the serial data rate and asynchronous character format (data bits; parity; stop bits). For implementation on the USB, the V.25ter commands that explicitly set these should be used:

- Port rate: encapsulated +IPR=<rate>, 6.2.10/V.25ter
- Character format: encapsulated +ICF=<format>,<parity>, 6.2.11/V.25ter.

For example:

AT+IPR=38400;+ICF=5,0 38400 bit/s, with 7 data bits, odd parity, and 1 stop bit

AT+IPR=19200;+ICF=3 19200 bit/s, with 8 bits data, no parity, and 1 stop bit

Note: On the USB there is no physical TXD lead to automatically detect the data rate and format, so +ICF=0 cannot be used. The communication device may ignore this parameter if it is set to 0.

C.3 Out-of-band Lead Control

TIA-617 (or ITU-T V.ib) contain means to encapsulate out-of-band leads in-band. These means should be used to represent serial out-of-band leads, if required by the communications application software. These means are controlled by the AT+IBM command (8.3/TIA-617). Output leads and input leads are encapsulated in the default endpoint (7.2.1) or the interrupt endpoint (7.2.3) respectively.

These commands are implemented as two character sequences: <EM><command>. <EM> is 19h. The commands for the most common out-of-band control leads are listed in the following table.

Table C.1: In-Band Serial Lead Representation

V.24 cct	Label	Source	Turn Off (hex)	Turn On (hex)	Description
105	RTS	Host	19, 42	19, 43	Request To Send
106	CTS	Device	19, 62	19, 63	Clear To Send
107	DSR	Device	19, 64	19, 65	Data Set Ready
108/2	DTR	Host	19, 44	19, 45	Data Terminal Ready
109	RLSD	Device	19, 66	19, 67	Received Line Signal Detect
125	RI	Device	19, 6A	19, 6B	Ring Indication
133	RFR	Host	19, 46	19, 47	Ready For Receiving

The TIA-617 command <EM><poll> (19h, 5Eh) may be used to interrogate the current state of the input leads. This is equivalent to issuing a read to the MSR of a 16450/16550 UART.